A New Pivoting and Iterative Text Detection Algorithm for Biomedical Images: Appendix A

Algorithm Details for our Pivoting Text Detection Algorithm

Inspired by the classical histogram analysis based text region detection methods ([2, 1]), we describe a procedure for locating text regions in an image through analyzing both the vertical and horizontal projection histograms of an image:

- **Input** An input image $\mathcal{I}$ and a specified rectangular region $\mathcal{R} = (left, right, top, bottom)$ inside the region of $\mathcal{I}$.
- **Function of the Procedure** To detect all the text regions inside the interior region $\mathcal{R}$ of the input image $\mathcal{I}$.
- **Output** A collection of text regions $\{\mathcal{R}_j^i\}$ so detected where each $\mathcal{R}_j^i$ is a text region detected from within the region $\mathcal{R}$ of the input image $\mathcal{I}$.

We can formally state the above procedure in the form of (1).

$$\text{Text Region Detection Procedure} : \mathcal{I}, \mathcal{R} \rightarrow \{\mathcal{R}_j^i\}. \tag{1}$$

We will now look at the details of our text detection procedure.

1. First, we convert the input image $\mathcal{I}$ into black and white if it is originally a color image. We then apply a 3x3 median filter to blur the

image background in order to make our text detection procedure less sensitive to image noise.

2. Next, we detect edges in the converted black and white image. Currently, we use the classical Sobel operator for this purpose due to its simplicity and satisfying performance in our experiments. Other edge detectors, such as Canny and Canny-Deriche edge detectors, can also be used without noticeably affecting the overall performance of our algorithm. We call the resultant image from this step the edge image of the original input image $\mathcal{I}$, which is denoted as $\widehat{\mathcal{I}}$.

3. We then compute the vertical projection for each pixel in the edge image $\widehat{\mathcal{I}}$ to derive $\widehat{\mathcal{I}}$'s horizontal histogram. More concretely, given the width w and height v (both in pixels) of $\widehat{\mathcal{I}}$, the horizontal projection histogram of the edge image $\widehat{\mathcal{I}}$ is denoted as $\mathcal{H}_{\mathrm{h}}(i)$ $(i = 1, \cdots, w)$ where $\mathcal{H}_{\mathrm{h}}(i)$ records the number of edge pixels on the vertical line that stays i pixels away from the left boundary of the image, i.e. $\mathcal{H}_{\mathrm{h}}(i) = |\{pixel(i, y)|pixel(i, y) \text{ is an edge pixel in } \widehat{\mathcal{I}}; y = 1, \cdots, v\}|$. Here $pixel(x, y)$ denotes the pixel whose horizontal and vertical coordinates are x and y respectively; and $|\mathbf{X}|$ returns the cardinality of the set $\mathbf{X}$. The overall horizontal histogram of the edge image $\widehat{\mathcal{I}}$ is thus represented as a w dimensional vector in the form of $\mathcal{H}_{\mathrm{h}} \triangleq [\mathcal{H}_{\mathrm{h}}(1), \cdots, \mathcal{H}_{\mathrm{h}}(w)]$.

4. We then segment the derived horizontal projection histogram $\mathcal{H}_{\mathrm{h}}$ according to a preset segmentation threshold τ_{h}. To carry out this segmentation, we first derive a binary sequence $\mathcal{B}_{\mathrm{h}}$ according to the horizontal projection histogram $\mathcal{H}_{\mathrm{h}}$. Here we define $\mathcal{B}_{\mathrm{h}}$ to be a w dimen-

sional vector in the form of $\mathcal{B}_{\mathrm{h}} \triangleq [\mathcal{B}_{\mathrm{h}}(1), \cdots, \mathcal{B}_{\mathrm{h}}(w)]$. For each $\mathcal{B}_{\mathrm{h}}(i)$, it is derived as follows:

$$\mathcal{B}_{\mathrm{h}}(i) \triangleq \begin{cases} 1 & \text{if } \mathcal{H}_{\mathrm{h}}(i) \geq \tau_h; \\ 0 & \text{otherwise.} \end{cases} \quad (i = 1, \cdots, w) \tag{2}$$

We then detect all the segments of consecutive 1's in $\mathcal{B}_{\mathrm{h}}$ and denote the resultant sequence of segments as $Seg_1, \cdots, Seg_n$ where we assume there are n such resulting segments. Here Seg_i corresponds to the i-th segment of consecutive 1's in $\mathcal{B}_{\mathrm{h}}$. For each such segment Seg_i, we represent its left and right boundaries in the binary sequence $\mathcal{B}_{\mathrm{h}}$ as $left_i$ and $right_i$ respectively. That is, Seg_i corresponds to the block of consecutive 1's which starts at the $left_i$-th component in $\mathcal{B}_{\mathrm{h}}$ and ends at the end_i-th component in $\mathcal{B}_{\mathrm{h}}$. It is easy to see that $right_{i-1} < left_i$ as otherwise Seg_{i-1} should have been joined with Seg_i. Also, if any resultant segment's width is less than 3 pixels apart, i.e. $left_i - right_{i-1} < 3$, we will eliminate this segment, as such a segment probably correlates to an edge or a boundary in the input image rather than a text region since with this narrow width, texts are unlikely to be eligible.

5. For each segment Seg_i obtained from the previous step, we can locate a rectangular sub region $\mathcal{R}(i)$ in the edge image $\widehat{\mathcal{I}}$. The left, right, top, bottom boundaries of the region correspond to the lines $x = left_i$, $x = right_i$, $y = 1$, and $y = v$ in the image $\widehat{\mathcal{I}}$ respectively. And all the pixels falling between these boundaries constitute the region $\mathcal{R}(i)$, which is denoted as $\mathcal{R}(i) \triangleq \{pixel(x, y) | left_i \leqslant x \leqslant right_i; 1 \leqslant y \leqslant v\}$. For each so located region $\mathcal{R}(i)$, we then derive its vertical projection histogram, which is denoted as $\mathcal{H}_{\mathrm{v}}^i$ where the subscript v indicates

it is a vertical histogram and the superscript i indicates this vertical histogram corresponds to the region $\mathcal{R}(i)$. Such a vertical histogram $\mathcal{H}_{\mathrm{V}}^i$ is represented as a v dimensional vector in the form of $\mathcal{H}_{\mathrm{V}}^i \triangleq [\mathcal{H}_{\mathrm{V}}^i(1), \cdots, \mathcal{H}_{\mathrm{V}}^i(v)]$ where $\mathcal{H}_{\mathrm{V}}^i(j)$ records the number of edge pixels on the horizontal line which stays j pixels above the bottom of the image, i.e. $\mathcal{H}_{\mathrm{V}}^i(j) = |\{pixel(x,j)|pixel(x,j) \text{ is an edge pixel in } \widehat{\mathcal{I}}; left_i \leqslant x \leqslant right_i\}|$. This way of deriving the vertical histogram vector $\mathcal{H}_{\mathrm{V}}^i$ is very similar to the process for deriving the horizontal histogram vector $\mathcal{H}_{\mathrm{h}}$ as examined earlier in step 3.

6. Once the vertical projection histogram $\mathcal{H}_{\mathrm{V}}^i$ has been derived, we can then segment the image region $\mathcal{R}(i)$ following a similar routine as employed in step 4 in the above. That is, we first derive a binary sequence $\mathcal{B}_{\mathrm{V}}^i \triangleq [\mathcal{B}_{\mathrm{V}}^i(1), \cdots, \mathcal{B}_{\mathrm{V}}^i(v)]$ according to $\mathcal{H}_{\mathrm{V}}^i$ as follows:

$$\mathcal{B}_{\mathrm{V}}^i(j) \triangleq \begin{cases} 1 & \text{if } \mathcal{H}_{\mathrm{V}}^i(j) \geq \tau_{\mathrm{V}}; \\ 0 & \text{otherwise.} \end{cases} \quad (j = 1, \cdots, v) \tag{3}$$

where τ_{V} is a pre-selected segmentation threshold.

Our algorithm then detects segments of consecutive 1's in $\mathcal{B}_{\mathrm{V}}^i$. The resultant sequence of such segments are denoted as $Seg_1^i, \cdots, Seg_{m_i}^i$ assuming there are m_i segments of consecutive 1's detected from $\mathcal{B}_{\mathrm{V}}^i$ in total. For each Seg_j^i, the i-th segment of consecutive 1's in B_v^i, we represent its left and right boundaries as $bottom_j^i$ and top_j^i respectively. That is, Seg_j^i corresponds to the block of consecutive 1's which starts at the $bottom_j^i$-th component in $\mathcal{B}_{\mathrm{V}}^i$ and ends at the top_j^i-th component in $\mathcal{B}_{\mathrm{V}}^i$. It is easy to see that $top_{j-1}^i < bottom_j^i$ as otherwise the two segments Seg_{j-1}^i and Seg_j^i should have been merged together. Similar to the small segment elimination process in step 4, if any resultant segment's height is less than 3 pixels apart, i.e. $bottom^i_j - top^i_{j-1} < 3$, we will eliminate this segment, as such a segment probably correlates to an edge or a boundary in the input image rather than a text region since with this narrow height, texts are unlikely to be eligible.

7. Every pair of the segments Seg_i and Seg^i_j $(j = 1, \cdots, m_i)$ derived in step 4 and 6 in the above jointly defines a rectangular region $\mathcal{R}^i_j$ inside the original image $\mathcal{I}$, whose left, right, top, bottom boundaries correspond to the lines $x = left_i$, $x = right_i$, $y = bottom^i_j$ and $y = top^i_j$ in $\mathcal{I}$ respectively. That is, $\mathcal{R}^i_j \triangleq \{pixel(x, y) | left_i \leqslant x \leqslant right_i; bottom^i_j \leqslant y \leqslant top^i_j\}$. Each such region serves as a candidate text region. For every $\mathcal{R}^i_j$, we compute a corresponding minimum coverage bounding box, which is denoted as $\mathcal{X}^i_j$. Initially, the boundaries of the bounding box $\mathcal{X}^i_j$ are set as the boundaries of the rectangular region $\mathcal{R}^i_j$. We then optimize positions of these boundaries through a two-stage expansion and shrinking process. In the first stage, the bounding box will be minimally expanded so that all the edge pixels which are connected to at least one edge pixel inside the text region $\mathcal{R}^i_j$ will be covered by the expanded bounding box. And then in the second stage, the bounding box will be maximally shrunk so that the area of the bounding box is minimized without excluding any edge pixels originally contained in the region $\mathcal{R}^i_j$. After this two-stage process searching for optimal boundary positions for $\mathcal{R}^i_j$, we add the bounding box region $\mathcal{X}^i_j$ into the result text region collection $\{\mathcal{R}\}$. All the text regions so derived constitute the result text region set $\mathcal{R}$, which are detected from the interior region

$\mathcal{R}$ of the input image $\mathcal{I}$.

[1] Lienhart, R. and Wernicke, A. (2002). Localizing and segmenting text in images and videos. *IEEE Transactions on Circuits and Systems for Video Technology*, **12**(4), 256–268.

[2] Wu, V., Manmatha, R., and Riseman, E. M. (1999). Textfinder: an automatic system to detect and recognize text in images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **21**(11), 1224–1229.